Amendments to the Claims

Please amend claims to be as follows.

1. (currently amended) A method of cross-file inlining during [[a]] compilation of a program, ~~the method comprising determining~~ wherein which files to open and close is determined based on affinity weightings between the files, wherein the affinity weightings depend on a number of potential inlines between the files, the method comprising: performing an iterative process including (i) choosing an edge in an inline affinity graph with a highest affinity weighting, (ii) retrieving and opening source files corresponding to the chosen edge, and (iii) updating the inline affinity graph; and continuing the iterative process until the affinity weightings of all edges in the inline affinity graph go to zero.

2. (currently amended) A method of compiling a computer program from a plurality of files of source code, the method comprising:

an inline analysis to determine which call sites in the plurality of files to inline, the inline analysis including performance of an iterative process including (i) choosing an edge in an inline affinity graph with a highest affinity weighting, (ii) retrieving and opening source files corresponding to the chosen edge, and (iii) updating the inline affinity graph, and continuation of the iterative process until the affinity weightings of all edges in the inline affinity graph go to zero; and

an inline transformation to perform said inlining. ~~within currently opened files, including determining which files to open and close in dependence on affinity weightings between the files.~~

3. (currently amended) The method of claim 2, wherein ~~affinity weightings are representable by an inline affinity graph whose~~ nodes in the inline affinity graph correspond to files and [[whose]] edges in the inline affinity graph correspond to a number of potential inlines across corresponding files.

4. (canceled)

5. (canceled)

6. (original) The method of claim 3, wherein an inline dependence for a call site is maintained including information as to a set of call sites that the call site depends upon.

7. (original) The method of claim 6, wherein inline dependencies are representable by an inline dependence graph.

8. (original) The method of claim 7, further comprising:
dynamically updating the inline dependence graph after inlinings within currently opened files are done.

9. (currently amended) An apparatus for compiling a program utilizing cross-file inlining, the apparatus comprising: a processor for executing instructions; a memory system for storing said instructions and data; and processor-executable instructions which determine[[s]] which files to open and close based on affinity weightings between the files, wherein the affinity weightings depend on a number of potential inlines between the files, the processor-executable instructions comprising instructions to perform an iterative process including (i) choosing an edge in an inline affinity graph with a highest affinity weighting, (ii) retrieving and opening source files corresponding to the chosen edge, and (iii) updating the inline affinity graph; and

instructions to continue the iterative process until the affinity weightings of all edges in the inline affinity graph go to zero.

10. (currently amended) An apparatus for compiling a computer program from a plurality of files of source code, the apparatus comprising:

a processor for executing instructions;

a memory system for storing said instructions and data;

processor-executable instructions for an analyzer module, the analyzer module being configured to determine which call sites in the plurality of files to inline, the analyzer module being further configured to perform an iterative process including (i) choosing an edge in an inline affinity graph with a highest affinity weighting, (ii) retrieving and opening source files corresponding to the chosen edge, and (iii) updating the inline affinity graph, and to continue the iterative process until the affinity weightings of all edges in the inline affinity graph go to zero; and

processor-executable instructions for a transformer module, the transformer being configured to perform said inlining ~~within currently opened files, including determining which files to open and close in dependence on an affinity weighting between the files~~.

11. (currently amended) The apparatus of claim 10, wherein ~~affinity weightings are representable by an inline affinity graph whose~~ nodes in the inline affinity graph correspond to files and [[whose]] edges in the inline affinity graph correspond to affinity weightings between files, wherein the affinity weightings between files depend at least upon the number of inlines between the files.

12. (canceled)

13. (canceled)

14. (original) The apparatus of claim 11, wherein an inline dependence for a call site is maintained including information as to a set of call sites that the call site depends upon.

15. (original) The apparatus of claim 12, wherein inline dependencies are representable by an inline dependence graph.

16. (canceled)

17. (currently amended) A computer program product comprising a computer-usable medium having computer-readable code embodied therein, the computer program product being compiled from a plurality of files of source code using an inline analyzer which determines which call sites in the plurality of files to inline, the inline analyzer performing an iterative process including (i) choosing an edge in an inline affinity graph with a highest affinity weighting, (ii) retrieving and opening source files corresponding to the chosen edge, and (iii) updating the inline affinity graph, and which continues the iterative process until the affinity weightings of all edges in the inline affinity graph go to zero, and an inline transformer which performs [[function]] said inlining ~~within currently opened files and determines which files to open and close in dependence on an affinity weighting between the files~~.